What is Machine Learning?

What is Machine Learning?

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:

Supervised learning and Unsupervised learning.

# Supervised Learning

## Supervised Learning

In supervised learning, we are given a data set **and already know what our correct output should look like**, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into **"regression" and "classification**" problems. In a regression problem, we are trying to predict results within **a continuous** output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in **a discrete** output. In other words, we are trying to map input variables into discrete categories.

# Unsupervised Learning

## Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

**Example:**

Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a [cocktail party](https://en.wikipedia.org/wiki/Cocktail_party_effect)).

# Model Representation

To establish notation for future use, we’ll use x^{(i)}*x*(*i*) to denote the “input” variables (living area in this example), also called input features, and y^{(i)}*y*(*i*) to denote the “output” or target variable that we are trying to predict (price). A pair (x^{(i)} , y^{(i)} )(*x*(*i*),*y*(*i*)) is called a training example, and the dataset that we’ll be using to learn—a list of m training examples {(x^{(i)} , y^{(i)} ); i = 1, . . . , m}(*x*(*i*),*y*(*i*));*i*=1,...,*m*—is called a training set. Note that the superscript “(i)” in the notation is simply an index into the training set, and has nothing to do with exponentiation. We will also use X to denote the space of input values, and Y to denote the space of output values. In this example, X = Y = ℝ.

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like this:



When the target variable that we’re trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

# Cost Function

We can measure the accuracy of our hypothesis function by using a **cost function**. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's.

J(\theta\_0, \theta\_1) = \dfrac {1}{2m} \displaystyle \sum \_{i=1}^m \left ( \hat{y}\_{i}- y\_{i} \right)^2 = \dfrac {1}{2m} \displaystyle \sum \_{i=1}^m \left (h\_\theta (x\_{i}) - y\_{i} \right)^2*J*(*θ*0​,*θ*1​)=2*m*1​*i*=1∑*m*​(*y*^​*i*​−*yi*​)2=2*m*1​*i*=1∑*m*​(*hθ*​(*xi*​)−*yi*​)2

# Gradient Descent

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields \theta\_0*θ*0​ and \theta\_1*θ*1​ (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put \theta\_0*θ*0​ on the x axis and \theta\_1*θ*1​ on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction in which the step is taken is determined by the partial derivative of J(\theta\_0,\theta\_1)*J*(*θ*0​,*θ*1​). Depending on where one starts on the graph, one could end up at different points. The image above shows us two different starting points that end up in two different places.

The gradient descent algorithm is:

repeat until convergence:

\theta\_j := \theta\_j - \alpha \frac{\partial}{\partial \theta\_j} J(\theta\_0, \theta\_1)*θj*​:=*θj*​−*α*∂*θj*​∂​*J*(*θ*0​,*θ*1​)

where

j=0,1 represents the feature index number.

At each iteration j, one should simultaneously update the parameters \theta\_1, \theta\_2,...,\theta\_n*θ*1​,*θ*2​,...,*θn*​. Updating a specific parameter prior to calculating another one on the j^{(th)}*j*(*th*) iteration would yield to a wrong implementation.



# Gradient Descent Intuition

In this video we explored the scenario where we used one parameter \theta\_1*θ*1​ and plotted its cost function to implement a gradient descent. Our formula for a single parameter was :

Repeat until convergence:

|  |
| --- |
| \theta\_1:=\theta\_1-\alpha \frac{d}{d\theta\_1} J(\theta\_1)*θ*1​:=*θ*1​−*αdθ*1​*d*​*J*(*θ*1​) |

Regardless of the slope's sign for \frac{d}{d\theta\_1} J(\theta\_1)*dθ*1​*d*​*J*(*θ*1​), \theta\_1*θ*1​ eventually converges to its minimum value. The following graph shows that when the slope is negative, the value of \theta\_1*θ*1​ increases and when it is positive, the value of \theta\_1*θ*1​ decreases.



On a side note, we should adjust our parameter \alpha*α* to ensure that the gradient descent algorithm converges in a reasonable time. Failure to converge or too much time to obtain the minimum value imply that our step size is wrong.



### **How does gradient descent converge with a fixed step size \alpha*α*?**

The intuition behind the convergence is that \frac{d}{d\theta\_1} J(\theta\_1)*dθ*1​*d*​*J*(*θ*1​) approaches 0 as we approach the bottom of our convex function. At the minimum, the derivative will always be 0 and thus we get:

|  |
| --- |
| \theta\_1:=\theta\_1-\alpha \* 0*θ*1​:=*θ*1​−*α*∗0 |

